Direct manipulation systems offer the satisfying experience of operating on visible objects. The computer becomes transparent, and users can concentrate on their tasks.

Direct Manipulation: A Step Beyond Programming Languages

Ben Shneiderman, University of Maryland

> Leibniz sought to make the form of a symbol reflect its content. "In signs," he wrote, "one sees an advantage for discovery that is greatest when they express the exact nature of a thing briefly and, as it were, picture it; then, indeed, the labor of thought is wonderfully diminished."
>
> Frederick Kreiling, "Leibniz," *Scientific American*, May 1968

Certain interactive systems generate glowing enthusiasm among users—in marked contrast with the more common reaction of grudging acceptance or outright hostility. The enthusiastic users' reports are filled with positive feelings regarding

- mastery of the system,
- competence in the performance of their task,
- ease in learning the system originally and in assimilating advanced features,
- confidence in their capacity to retain mastery over time,
- enjoyment in using the system,
- eagerness to show it off to novices, and
- desire to explore more powerful aspects of the system.

These feelings are not, of course, universal, but the amalgam does convey an image of the truly pleased user. As I talked with these enthusiasts and examined the systems they used, I began to develop a model of the features that produced such delight. The central ideas seemed to be visibility of the object of interest; rapid, reversible, incremental actions; and replacement of complex command language syntax by direct manipulation of the object of interest—hence the term "direct manipulation."

Examples of direct manipulation systems

No single system has all the attributes or design features that I admire—that may be impossible—but those described below have enough to win the enthusiastic support of many users.

Display editors. "Once you've used a display editor, you'll never want to go back to a line editor. You'll be spoiled." This reaction is typical of those who use full-page display editors, who are great advocates of their systems over line-oriented text editors. I heard similar comments from users of stand-alone word processors such as the Wang system and from users of display editors such as EMACS on the MIT/Honeywell Multics system or "vi" (for visual editor) on the Unix system. A beaming advocate called EMACS "the one true editor."

Roberts[1] found that the overall performance time of display editors is only half that of line-oriented editors, and since display editors also reduce training time, the evidence supports the enthusiasm of display editor devotees. Furthermore, office automation evaluations consistently favor full-page display editors for secretarial and executive use.

The advantages of display editors include

Display of a full 24 to 66 lines of text. This full display enables viewing each sentence in context and simplifies reading and scanning the document. By contrast, the

---

A portion of this article was derived from the author's keynote address at the NYU Symposium on User Interfaces, "The Future of Interactive Systems and the Emergence of Direct Manipulation," published in *Human Factors in Interactive Computer Systems*, Y. Vassiliou, ed., Ablex Publishing Co., Norwood, N.J., 1983.

0018-9162/83/0800-0057$01.00

one-line-at-a-time view offered by line editors is like seeing the world through a narrow cardboard tube.

Display of the document in its final form. Eliminating the clutter of formatting commands also simplifies reading and scanning the document. Tables, lists, page breaks, skipped lines, section headings, centered text, and figures can be viewed in the form that will be printed. The annoyance and delay of debugging the format commands is eliminated because the errors are immediately apparent.

Cursor action that is visible to the user. Seeing an arrow, underscore, or blinking box on the screen gives the operator a clear sense of where to focus attention and apply action.

Cursor motion through physically obvious and intuitively natural means. Arrow keys or devices such as a mouse, joystick, or graphics tablet provide natural physical mechanisms for moving the cursor. This is in marked contrast with commands such as UP 6, which require an operator to convert the physical action into correct syntactic form and which may be difficult to learn, hard to recall, and a source of frustrating errors.

Labeled buttons for action. Many display editors have buttons etched with commands such as INSERT, DELETE, CENTER, UNDERLINE, SUPERSCRIPT, BOLD, or LOCATE. They act as a permanent menu selection display, reminding the operator of the features and obviating memorization of a complex command-lan-

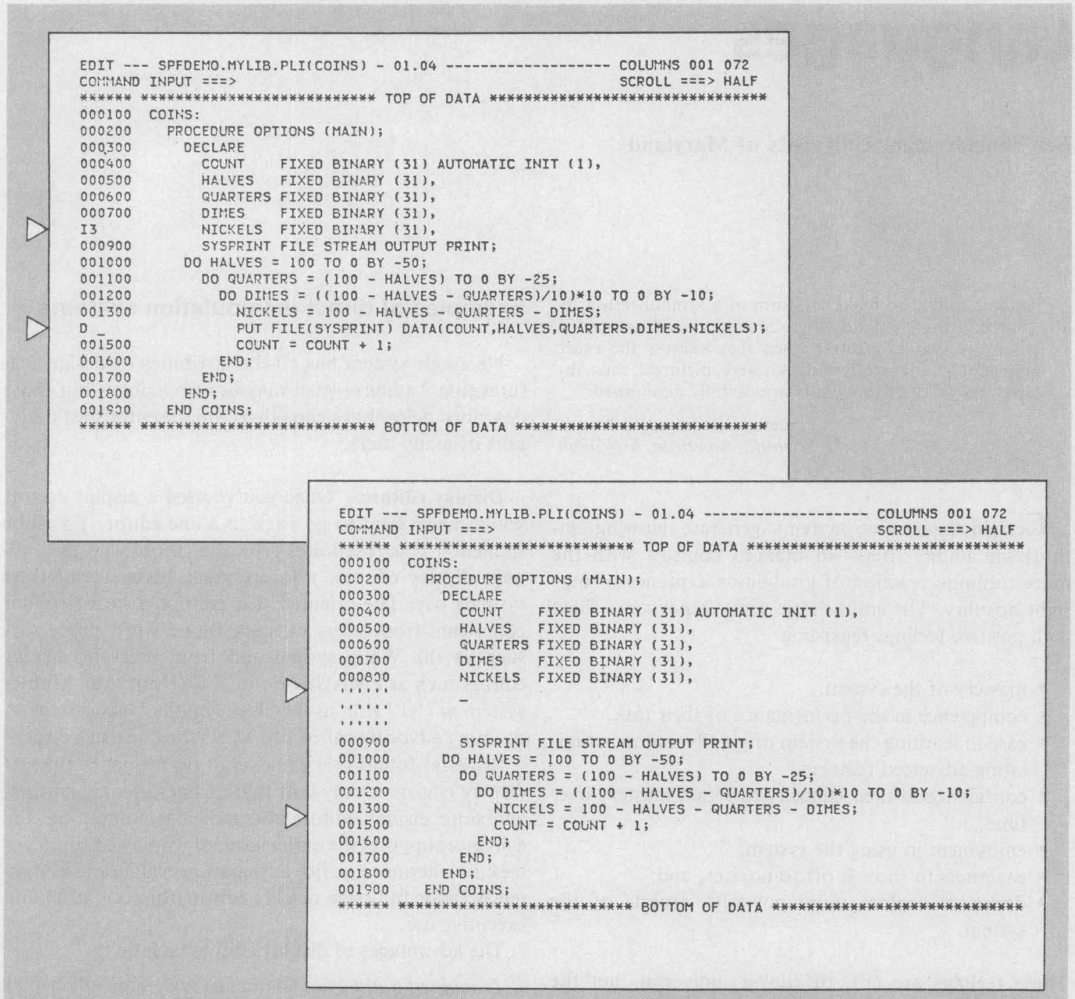


Figure 1. This example from the IBM SPF display editor shows 19 lines of a PL/I program. The commands to insert three lines (I3) and to delete one line (D or D1) are typed on the appropriate lines in the first screen display. Pressing ENTER causes commands to be executed and the cursor to be placed at the beginning of the inserted line. New program statements can be typed directly in their required positions. Control keys move the cursor around the text to positions where changes are made by overstriking. A delete key causes the character under the cursor to be deleted and the text to the left to be shifted over. After pressing an insert key, the user can type text in place. Programmed function keys allow movement of the window forwards, backwards, left, and right over the text. (Examples courtesy of IBM.)

guage syntax. Some editors provide basic functionality with only 10 or 15 labeled buttons, and a specially marked button may be the gateway to advanced or infrequently used features offered on the screen in menu form.

Immediate display of the results of an action. When a button is pressed to move the cursor or center the text, the results appear on the screen immediately. Deletions are apparent at once, since the character, word, or line is erased and the remaining text rearranged. Similarly, insertions or text movements are shown after each keystroke or function button press. Line editors, on the other hand, require a print or display command before the results of a change can be seen.

Rapid action and display. Most display editors are designed to operate at high speeds: 120 characters per second (1200 baud), a full page in a second (9600 baud), or even faster. This high display rate coupled with short response time produces a thrilling sense of power and speed. Cursors can be moved quickly, large amounts of text can be scanned rapidly, and the results of commands can be shown almost instantaneously. Rapid action also reduces the need for additional commands, thereby simplifying product design and decreasing learning time. Line editors operating at 30 characters per second with three- to eight-second response times seem sluggish in comparison. Speeding up line editors adds to their attractiveness, but they still lack features such as direct overtyping, deletion, and insertion.

Easily reversible commands. Mistakes in entering text can be easily corrected by backspacing and overstriking. Simple changes can be made by moving the cursor to the problem area and overstriking, inserting, or deleting characters, words, or lines. A useful design strategy is to include natural inverse operations for each operation. Carroll[2] has shown that congruent pairs of operations are easy to learn. As an alternative, many display editors offer a simple UNDO command that cancels the previous command or command sequence and returns the text to its previous state. This easy reversibility reduces user anxiety about making mistakes or destroying a file.

The large market for display editors generates active competition, which accelerates evolutionary design refinements. Figure 1 illustrates the current capabilities of an IBM display editor.

Visicalc. Visicorp's innovative financial forecasting program, called Visicalc, was the product of a Harvard MBA student, who was frustrated by the time needed to carry out multiple calculations in a graduate business course. Described as an "instantly calculating electronic worksheet" in the user's manual, it permits computation and display of results across 254 rows and 63 columns and is programmed without a traditional procedural control structure. For example, positional declarations can prescribe that column 4 displays the sum of columns 1 through 3; then every time a value in the first three columns changes, the fourth column changes as well. Complex dependencies among manufacturing costs, distribution costs, sales revenue, commissions, and profits can be stored for several sales districts and months so that the impact of changes on profits is immediately apparent.

Since Visicalc simulates an accountant's worksheet, it is easy for novices to comprehend. The display of 20 rows and up to nine columns, with the provision for multiple windows, gives the user sufficient visibility to easily scan information and explore relationships among entries (see Figure 2). The command language for setting up the worksheet can be tricky for novices to learn and for infrequent users to remember, but most users need learn only the basic commands. According to Visicalc's distributor, "It jumps," and the user's delight in watching this propagation of changes cross the screen helps explain its appeal.

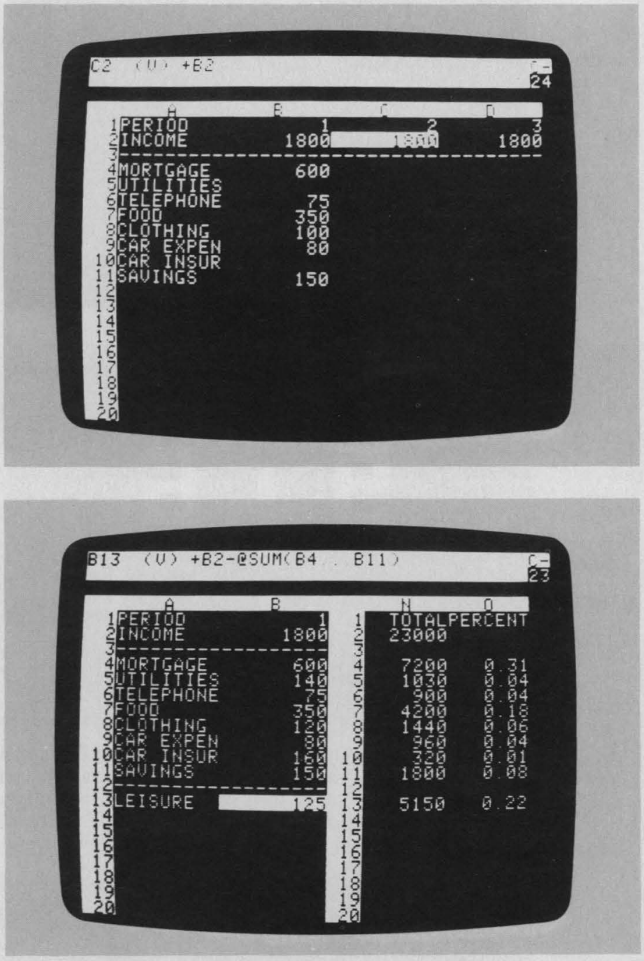
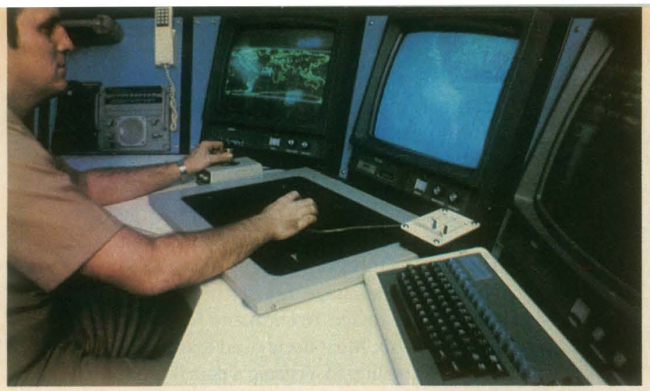
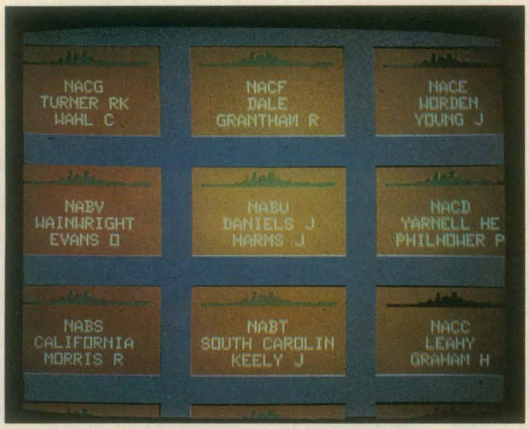


Figure 2. This simple Visicalc program display (top) shows four columns and 20 rows of home budget information. The cursor, an inverse video light bar controlled by key presses, is in position C2. The top command line shows that C2 is a value (as opposed to a text string) that has been set up to have the same value as position B2.

The second display (above) shows two windows over the home budget data with row sums to the right. The last row shows leisure dollar amounts, which are established by the top command line formula as the income minus the sum of expenses. A change to the income or expense values would immediately propagate to all affected values. (Displays reproduced by permission of Visicorp.)



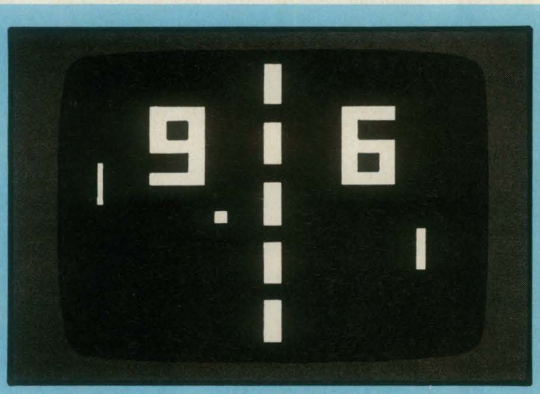


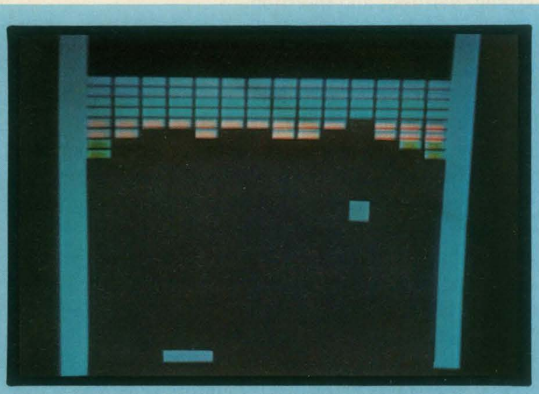
Spatial data management. The developers of the prototype spatial data management system[3] attribute the basic idea to Nicholas Negroponte of MIT.

In one scenario, a user seated before a color graphics display of the world zooms in on the Pacific to see markers for military ship convoys. Moving a joystick fills the screen with silhouettes of individual ships, which can be zoomed in on to display structural details or, ultimately, a full-color picture of the captain. (See Figure 3.)

In another scenario, icons representing different aspects of a corporation, such as personnel, organization, travel, production, or schedules, are shown on a screen. Moving the joystick and zooming in on objects takes users through complex "information spaces" or "I-spaces" to locate the item of interest. For example, when they select a department from a building floor

Figure 3. A spatial data management system has been installed on the aircraft carrier USS *Carl Vinson*. In the photo at top left, the operator has a world map on the left screen and a videodisc map of selected areas on the center screen. After some command selections with the data tablet and puck, the operator can zoom in on specific data such as the set of ships shown in the second photo. With further selections the operator can get detailed information about each ship, such as the length, speed, and fuel. (Photos courtesy of Computer Corporation of America.)





In 1971, about the only people playing video games were students in computer science laboratories. By 1973, however, millions of people were familiar with at least one video game—Pong (above left). A few years later came Breakout (above right), which, according to many designers, was the first true video game and the best one ever invented. Pong and other early games imitated real life, but Breakout could not have existed in any medium other than video. In the game, a single paddle directed a ball toward a wall of color bricks; contact made a brick vanish and changed the ball's speed.

When the first arcade video game, Computer Space, went on location in a Sears store, its joystick was torn off before the end of the first day. As a result, game designers have sought controls that were both easy to use and hard to destroy. Centipede (above left) uses simple controls—a trackball and one button. On the other hand, Defender (above right) has five buttons and a joystick; novice players are confused by these relatively complex controls and usually give up after a few seconds.

plan, individual offices become visible. Moving the cursor into a room brings the room's details onto the screen. If they choose the wrong room, they merely back out and try another. The lost effort is minimal, and no stigma is attached to the error.

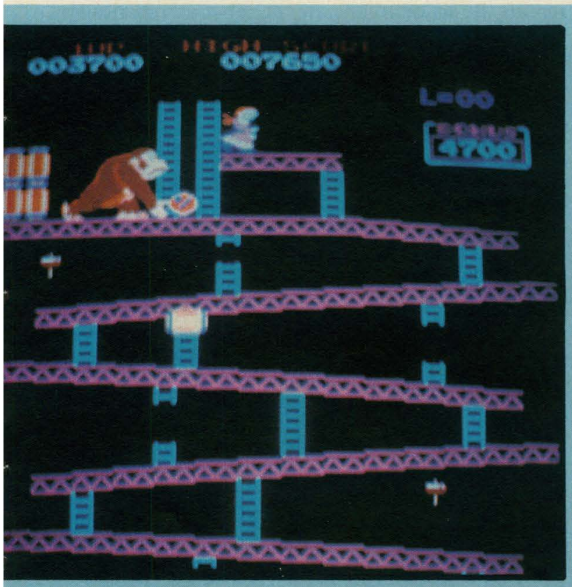
The success of a spatial data management system depends on the designer's skill in choosing icons, graphical representations, and data layouts that are natural and easily understood. Even anxious users enjoy zooming in and out or gliding over data with a joystick, and they quickly demand additional power and data.

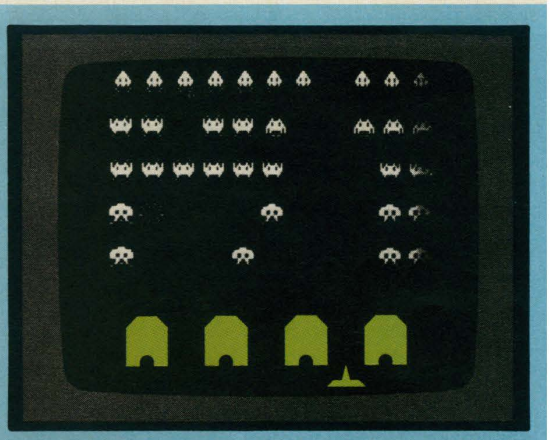
Video games. Perhaps the most exciting, well-engineered—certainly, the most successful—application of direct manipulation is in the world of video games. An early, but simple and popular, game called Pong required the user to rotate a knob, which moved a white rectangle on the screen. A white spot acted as a Ping-Pong ball, which ricocheted off the wall and had to be hit back by the movable white rectangle. The user developed skill involving speed and accuracy in placement of the "paddle" to keep the increasingly speedy ball from getting by, while the speaker emitted a ponging sound when the ball bounced. Watching someone else play for 30 seconds was all the training needed to become a competent novice, but many hours of practice were required to become a skilled expert.

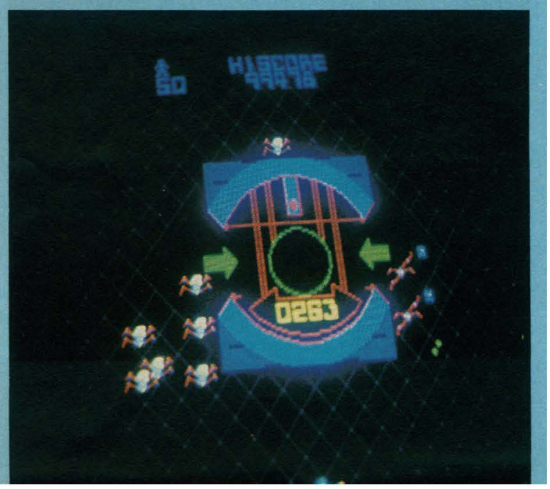
Contemporary games such as Missile Command, Donkey Kong, Pac Man, Tempest, Tron, Centipede, or Space Invaders are far more sophisticated in their rules, color graphics, and sound effects (see sidebar below and on facing page). The designers of these games have provided stimulating entertainment, a challenge for novices and experts, and many intriguing lessons in the human factors of interface design—somehow they have found a way to get people to put coins into the sides of computers. The strong attraction of these games contrasts markedly with the anxiety and resistance many users experience toward office automation equipment.

Because their fields of action are abstractions of reality, these games are easily understood—learning is by analogy. A general idea of the game can be gained by watching the on-line automatic demonstration that runs continuously on the screen, and the basic principles can be learned in a few minutes by watching a knowledgeable player. But there are ample complexities to entice many hours and quarters from experts. The range of skill accommodated is admirable.

The commands are physical actions, such as button presses, joystick motions, or knob rotations, whose results appear immediately on the screen. Since there is no syntax, there are no syntax error messages. If users move their spaceships too far left, then they merely use the natural inverse operation of moving back to the right. Error messages are unnecessary because the results of ac-







nkey Kong, Space Invaders, and Tron (clockwise from above) emplify the lively variety of video games now inviting the er's loose change. As of mid-1981, according to Steve Bloom, thor of *Video Invaders*, more than four billion quarters had en dropped into Space Invaders games around the rld—that's roughly "one game per earthling."

eo game photos reprinted courtesy of *IEEE Spectrum*. For a more complete ort on the topic, see "Video Games: The Electronic Big Bang" by Tekla ry, Carol Truxal, and Paul Wallich in *IEEE Spectrum*, Vol. 19, No. 12, Dec. 82, pp. 20-33.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.